BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Black Codher Bootcamp

NÍ YỌ

BLACK CODHER

CODING PROGRAMME

UNIT 4 - Session 8
React

Session 7 Summary

1. What is an API?
2. RESTful APIs
3. Google Book API and how to connect to it
4. Fetch API
5. Bookcase App with APIs
6. Useful UI React libraries

Goals for Unit 4 - Session 8

Quick review of all concepts:

1. What is React
2. React.createElement
3. Create React App Command
4. What is NPM and Node.js
5. Testing with Jest
6. JSX Components
7. Class Components
8. Functional Components
9. Component Lifecycle
10. Properties (Props) Explained
11. State Explained
12. React Hooks
13. Deconstructing
14. Conditional Rendering
15. PropTypes and Defaults
16. Nested Components
17. Handling Events
18. Lifting State Up
19. React Routes
20. Fragments
21. React Forms

What is React?

What is React?

Library not a Framework

- React is a JavaScript library for building fast and interactive interfaces
- With React you compose user interfaces using small isolated pieces of code called "components"
- React mainly responds to changes in the **Virtual DOM** by updating the component in the real **DOM**
- Unlike Angular (which is a JavaScript framework), React is relatively light-weight and requires other libraries in order to work

BLACK
CODHER

React.createElement

React.createElement()

- The **react.CreateElement()** call is the underlying call used to build React applications

```
React.createElement(type,[props],[...children])
```

- The type can be a name, string or element

- The [**props**] are attributes of the type such as style, class and events

- The children are the nested inside the element, in this case the text ‘**Click To Reveal**’ is a child of the button element

- It's important to note that JavaScript **styles** use **camelCasing** as opposed to **kebab-casing**, so **background-color** becomes, background**C**olor!

create-react-app command

Creating-React-App

- **Create-React-App** is a command line tool that sets up a boiler-plate development environment with the latest JavaScript features.

- To create a project using **create-react-app** we ran the create-react-app command from the commandline

```
> npx create-react-app myfirstapp
```

- create-react-app creates a frontend build pipeline using Babel (babeljs.io) and Webpack

BLACK CODHER

CODING PROGRAMME

NPM and Node.js

What is NPM and Node.js?

- **npm** (node package manager) is the dependency/package manager you get out of the box when you install **Node.js**. It provides a way for developers to install packages and modules both globally and locally

- Installing a package with **NPM** will install the package dependencies in your local **node_modules** folder

What is Node.js!?

- Node is an **open source** runtime environment for executing **JavaScript** code outside of a browser window (on a server)

Testing with Jest

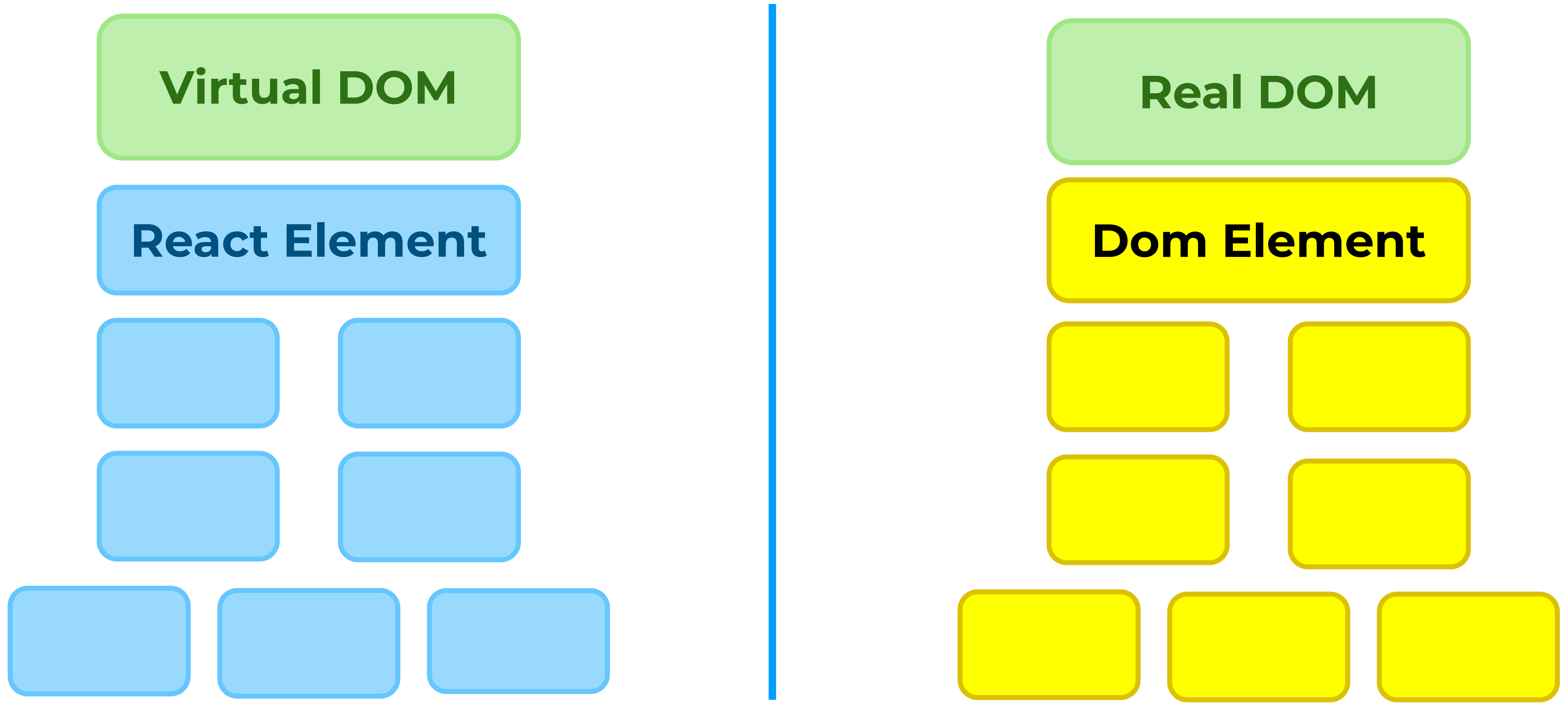
Testing React with Jest

- Jest is a JavaScript Testing Framework with a focus on simplicity. It can be used to test your React app
- The test runner also ‘watches’ the project files and re-runs when changes are made to the project files
- If running on the **create-react-app** the test code can be found in App.test.js
- More information on testing react can be found here https://create-react-app.dev/docs/running-tests

React Components

What is a Component?

- Components are at the heart of React applications
- React components are added in a tree structure to a Virtual Document Object Model (ReactDOM), which maps to a real DOM element
- With React you build several independent, isolated and reusable components that you can use to construct a user interface



JSX

JSX

- JSX stand for **J**avaScript e**X**tensible markup language.

- JSX is a syntax extension to JavaScript. It is optional and not required to use in React

- Each React element is a JavaScript object that you can store in a variable or pass around in your program. This can make your code more reusable

- React does not require JSX but it can be helpful as a visual aid when working with UI (User Interface) inside JavaScript code

- It’s important to note that JSX "transpiles" (translate and compiles) to plain JavaScript React.createElement() calls

BLACK CODHER

Class Components

Class Component

- Class components need a **render()** method which will return the React.Elements
- A class component takes in parameters, called props (short for properties)
- There are currently no plans to remove classes from React, however React creators do not recommend writing class components anymore in favour of **React Hooks**
- It's helpful to recognise the class component syntax and understand how to convert them to functional components

Functional Components

Functional Components

- Functional components also known as **stateless components** are components that are functions.
- One of the main differences between a **functional** component and **class** components is the syntax.
- functional components do not require you to extend from **React.Component**.
- They are normal JavaScript functions that take a **props** argument.
- They do not have a **render()** method. You return your react element directly as a return object of the method.

Component Lifecycle

Component Lifecycle

- Everything follows a lifecycle (e.g. animals, plants). They are born, they grow, and then they die.
- React Components also follows a similar cycle:

1. Components are created (**mounted** on the DOM).
2. The components grow (**updated**).
3. Components then die (**unmounted** on DOM).

- This is referred to as the **Component Lifecycle**.

Component Lifecycle

- The lifecycle methods are available from **class components** as components defined as classes currently provide more features
- Each of the stages described birth/mount, growth/update and death/unmount have a set of methods that can be overridden in the class
- The only method you must define in a React.Component subclass is called render() all other methods are optional

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about.

YELLOW - I have some questions but feel like I understand some things.

GREEN - I feel comfortable with everything you've said.

BLACK CODHER

CODING PROGRAMME

React Quiz

Handling Events

Handling Events

- Handling events with React is very similar to handling events on DOM elements with a few syntax changes

- Events on React elements use camel**C**asing rather than lower case (JavaScript) e.g. onChange instead of onchange

- Another difference is you cannot return false to prevent default behaviour you must call **preventDefault** explicitly

BLACK CODHER

Props

Props (Properties)

- **Props**, which is short for **properties** are arbitrary inputs accepted by functional components as arguments

- The props contains any **attribute** attached to the element when declared. For example, you can declare a Welcome component with the attributes:

```
const element = <Welcome firstname="Faith" lastname="Evans"/>;
```

- The props object will have the following attributes:

```
props {
  firstname : "Faith"
  lastname: "Evans"
}
```

Props (Properties)

- The property can then be accessed in the Welcome function using dot notation e.g. props.firstname

```
function Welcome(props) {
  return <h1>Hello, {props.firstname + " " + props.lastname}</h1>;
}
```

- Props can contain any JavaScript element, function or variable

- Props are read-only, so you **cannot** modify props inside a functional or class component

State

State

- React has another special built-in object called **state**, which allows components to create and manage their own data
- Unlike **props**, components cannot pass data with **state**, but they can create and manage it internally
- State is set in the constructor() of a component which is called only once when the component is created.
- State should not be modified directly but can be modified with a special method called setState()
- Changing the state of a React component will trigger a re-rendering of the component (Not the whole DOM)

React Hooks

React Hooks

- **Hooks** are a new addition in React 16.8 (released in Feb 2019).
- They let you use state and other React features without writing a class.
- Hooks are special functions that allow you to 'hook onto' React state and lifecycle features inside a **functional component**.
- Hooks are backward compatible, so, It is possible to convert your **class components** into **functional components** that use Hooks.
- React Hooks are completely **opt-in**, you can try Hooks in a few components without rewriting existing code.

Destructuring

Destructuring Assignment

- Objects allow us to create a single entity that holds data items by keys and arrays allow to hold data items in a collection.
- **Destructuring** assignment is a syntax in ES6 that allows us to unpack an array or objects into variables .
- It is great for reducing the complexity of code and shortening the dot notation needed to reference attributes in a complex objects.

Conditional Rendering

Conditional Rendering

- It's important to note that we **can't use if/else statements inside a functional JSX component declaration**.

- **Conditional logic** or **ternary operators** have to be used to short-circuit evaluation

e.g.

```
<p>by {authors ? authors.join(', '):"No Authors Listed"}</p>
```

or

```
{(props.stored==="library") && <h2>Suggested Reading</h2>}
```

Prop Types and Defaults

PropTypes

- As an App grows, they may be an increasing amount of bugs. Some of these may be down to type checking errors. To help with checking types you can import the **prop-types** library

- It contains validators that can help ensure your data is of the correct type/shape. Some of the validators are listed below

- It's possible to chain and nest the validators in one command e.g. `array.IsRequired`

- PropTypes exports a range of validators that can be used to make sure the data you receive is valid

Default PropTypes

- When an invalid value is provided for a prop, a warning will be shown in the JavaScript console.
- It's important to note that the application will still run but the warning will also be displayed in the browser console
- It is also possible to set a default property, this is useful for when the data returned is null or empty for some object

```
Book.defaultProps = {
   price: 'No price provided'
};
```

Nested Components

Nested Components

- A lot of the power of React is its ability to allow nesting of components

- Nesting components makes it easy and fast to separate UI elements and easy to inject **props** down to children based on the parent component's state

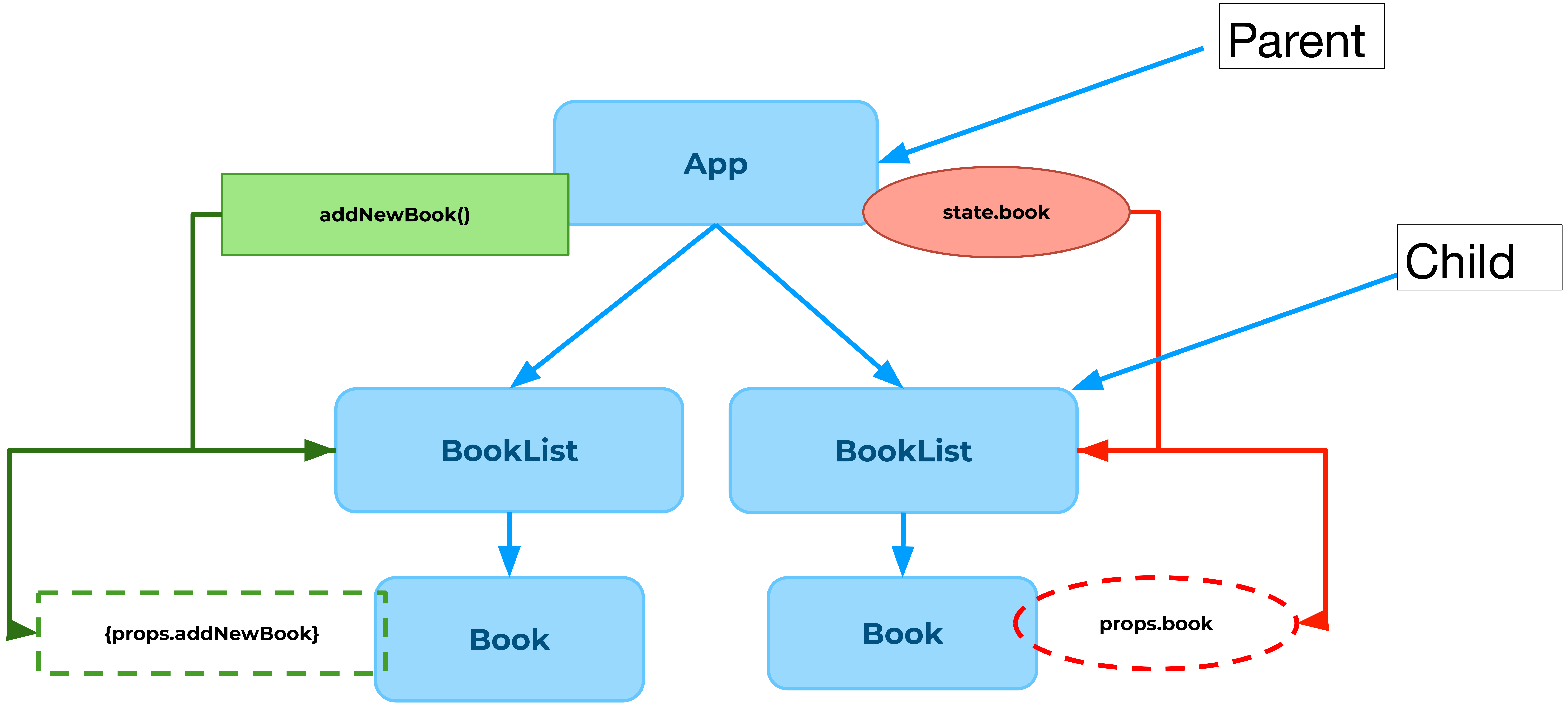
- There are a few ways to nest components:

1. Nesting using props.children (Containment)
2. Nesting with child components (Specialization)

Lifting State Up

Lifting State Up

- In React, data can be passed from parent to child and from child to parent



Lifting State Up

- To share a **state** between components the most common practice is to **lift state up** to a child component's most common ancestor. Removing **state** from the local component and **lifting it up** to the parent

- This is commonly done to help with sharing **state** across multiple components

- Each event handle is then passed down to the relevant child via the **props**

BLACK CODHER

CODING PROGRAMME

React Routes

React Routes

BLACK CODHER

- Routing is the ability to move between different parts of an application when a user enters a URL or clicks an element within the application (e.g. link, tab, button etc...)

- To add routes to the React application we can use a popular routing library called **react-router-dom**

- The **react-router-dom** is designed to be used for web applications

- To use the **react-router-dom** we will need to install the library using a package manager

```
> npm install react-router-dom
```

Fragments

Fragments

- A component can return only one base element however a common pattern in React is for a component to return multiple elements
- React.Fragment and the new, shorter syntax can solve this problem
- Fragment provides a base element that can be used to group a list of children without adding extra nodes to the DOM, reducing bloat
- Fragments declared with the explicit <React.Fragment> syntax may have keys. Key is the only attribute that can be passed to a Fragment

BLACK CODHER

CODING PROGRAMME

Form Management

Forms & Controlled Components

- HTML forms work differently from other DOM elements in React, because form elements naturally keep some internal state
- Forms have default HTML form behavior of browsing to a new page when the user submits the form. If you want this behavior in React, it just works
- However, to implement a Javascript function that handles the form submission we would use a technique called "**controlled components**"
- In HTML form elements such as **input**, **textarea** and **select** maintain their own state and update based on user input
- In React, state is kept in the state property and updated with the **setState** function or **useState** Hook

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about.

YELLOW - I have some questions but feel like I understand some things.

GREEN - I feel comfortable with everything you've said.

Summary of Session 8

Quick review of all concepts:

1. What is React
2. React.createElement
3. Create React App Command
4. What is NPM and Node.js
5. Testing with Jest
6. JSX Components
7. Class Components
8. Functional Components
9. Component Lifecycle
10. Properties (Props) Explained
11. State Explained
12. React Hooks
13. Deconstructing
14. Conditional Rendering
15. PropTypes and Defaults
16. Nested Components
17. Handling Events
18. Lifting State Up
19. React Routes
20. Fragments
21. React Forms

BLACK
CODHER

Homework

Finish the Bookcase Application

Build a full application for searching and storing books that you have read and want to read. The application will be called MyBookase.

Ensure your app has the following functionality:

1. Load an initial set of suggested books from a local data store (JSON file).

2. Add a book to a bookcase of books

3. Allow a user to navigate between a search screen and their bookcase of books. (Add routing)

4. On the search screen, the book generator will link to a third-party API provided by Google (Books APIv1: https://developers.google.com/books/docs/overview)

Building a React Application

5. Allow a user to view an about us page

6. Keep a count of the number of books in the bookcase and display in the document.title and on the page.
   You may wish to use useEffect to update the title on every app load.

7. Add a search bar form to the search page.

8. Search the API to display books relevant to the query

9. Allow a user to browse a library of books by book name, author or theme
   ***Optional* see https://developers.google.com/books/docs/v1/using#PerformingSearch under inauthor and subject**

10. Remove a book from a bookcase of books

Building a React Application

11. Add pagination (next and previous button)
    ***Optional* this will require storing the book data and page number in state, then only showing so many results per page.**

12. Advanced Pagination: Add numbered pages and result display